

Microsoft .NET

A C499 Presentation Report

**Authors:
Billy Yeung
Victor Salamon**

Instructor: Osmar Zaiane

Introduction

The best way to understand Microsoft .NET is to analyze its goals. It has been observed that today's computing shifts to distributed applications. This is mostly due to the impressive advent of cheap and relatively high-speed networking. Over the last few years, the cable and high-speed phone lines have brought Internet in the homes of much more users than it used to in the past. This combined with the effect of Moore's Law (predicting a doubling of processing power every eighteen months and a drop in price by a factor of two) makes distributed computing fully available to the common user for the first time.

There are many examples of distributed applications today. Napster uses a rich client talking to a directory service in the "Internet cloud," with all of the participating computers on the network acting as servers. Communication gadgets such as "Messenger" make use of web services to maintain "buddy list", schedule and personal information somewhere in the Internet cloud and communicates over the network directly with other clients in a peer-to-peer fashion. These distributed applications take advantage of the computing power at the edge of the network-on client machines-while providing services that are inherently connected. However, there is no common ground in developing these applications; this makes it difficult to produce applications in a manner that is as easy as producing them for the desktop.

The .NET technology is aimed at accelerating, unifying and integrating this next generation of distributed computing. Ideally, every service becomes a Web service, so that it can participate in the connected network. This applies to both applications and to resources like storage. Web services must be easy to aggregate and integrate so that developers can quickly and efficiently create applications. Users must have a simple and compelling experience working with Web services so that they will adopt the new applications and services.

Another goal is to have any application run on any device. Applications and data need to seamlessly flow from one device to another, in order to satisfy user's demands for running their applications on their favorite (mobile or not) devices. Thus, the platform on which .NET is going to run on is going to be a virtual one (Internet). Across this platform, software delivery will be a service in itself, in the sense that applications will be easily delivered into the user's device and integrated seamlessly.

What does it mean for users?

Have you ever been bothered by the fact that you can't have access to a specific application on your Palm, application that was originally developed for a desktop? Furthermore, don't you hate it when your friends have all these cool applications for their cell-phones, but you can't use them, because your phone's OS does not support them?

.NET is important to end users it frees them from the artificial constraints of hardware: user data will live on the Internet, not on the local machine. In addition, it can be accessed seamlessly from any desktop, laptop, cell phone, or PDA. It can even be interchanged between devices. Thus, it makes computers easier to use and far more functional.

Another declared goal of Microsoft .NET is creating a unified architecture for Internet services. Many web pages today offer different services that are more or less efficient. It happens many times that finding a specific product or information on a known company site is a frustrating and time-consuming activity. Even though the searched item is there, it is often hard to find it because the site might not be organized properly. .NET architecture will free users from the

limitations imposed by the data chaos that exists on the Web today. Users will have access to their data, however they want to view it, however they want to use it.

There is currently a market move tendency towards fully featured devices such as smart phones, PDAs and internet-ready televisions. Not only they have many features, but also their user interface is - or at least, is intended to be - easy to adapt to. However, the existence of a multitude of manufacturers implies an extreme heterogeneity in hardware and the software services they offer. This leads to a whole set of applications that overlap in functionality but have very different interfaces, installation and connectivity procedures.

.NET increases the reach of applications and enables the continual delivery of software. To the casual user, software is currently viewed as something that you have to install from a CD. A much more natural approach is to provide the software as a service-like caller ID or pay-per-view television-to subscribe to through a communications medium. This subscription should be enough to make the newly installed software integrate with your current setup, no matter what hardware it runs on. Microsoft's .NET philosophy says that software should integrate on your television or your cell phone just as easy as it would on your desktop.

What does it mean for business?

The driving force behind Microsoft .NET is a silent shift of focus from individual Web sites, applications, and devices to new constellations of computers, devices, and services that work together to deliver a solution that can serve a broader range and gives richer solutions. Under this scheme computers, devices, and services are able to communicate and collaborate with each other. Businesses will be able to offer their products and services in a way that lets customers embed them in their own electronic fabric.

To see what businesses can benefit from .NET, first consider some of the deficiencies in today's computing environment.

- The human –computer interaction mechanisms are mainly limited to keyboard and mouse for input, and monitor or paper for output. The communication between computer and humans is not giving a comfortable and sound experience to some users;
- User information is sited on the machine where he/she is using. Users cannot directly load their preferences or data file in any other machines (Previous solutions to this problem exist, such as Xdrive; however, this is not a unified, integrated solution.);
- It is hard to transform and integrate data from one application to another;
- Applications designed for particular devices - whether it's a PC, a pager, a cell phone, or a PDA - cannot be directly transferred to other devices.

.NET promises to solve these deficiencies by enabling access to all of a user's data and applications anywhere and from any device. In addition, .NET technology provides mechanism to enable applications linked in logical ways by extensive use of XML.

As stated above, .NET is a vision of making information available any time, any place, on any device. .NET enable users to interact with their computer and data through various input methods like handwriting, speech, and vision technologies. User's data is located in a common repository where they can access to their data anywhere on any machine. The common repository is connected to the Internet in order to satisfy this condition. For example, user's data will live securely on the Internet so that they can access it from their PCs at work and at home, from their cell phones or pagers, from their PDAs, and even from combination of page/cell phone/PDA-PC device that's on

the market horizon. Applications will be able to gracefully adapt the functionality they offer to the limitations (e.g. screen size) and opportunities presented by the device with which the user is interacting. As user profile and preferences is stored on the Internet, applications will be able to present an adaptive user interface on a user's behalf.

.NET makes users more productive. Since .NET enhances human computer interaction, people do not need to worry about how to interact with computers; or how to synchronize data from one application to another. Rather, people are lead to a place where they can focus on what to do with their computers to perform their tasks and accomplish their goals.

Businesses will thus benefit from dramatic increase of efficiency and productivity. As well, the cost to educate customers or employees on the user of any software would decrease, since the user interface is drawn from the user experience and preferences that is stored on the Internet. .NET brings employees, customers, data, and business applications into a coherent and intelligently interactive whole. ".NET promises a world of business without boundaries."

What does it mean for developers ?

Microsoft attempts to extend the current view of what an operating system is and expand it beyond the limits of a device, regardless of whether this is a desktop or mobile one. The .NET technology allows distributing and integrating operating-system specific services over the Internet. Ultimately, this will allow developers to create programs that are independent of the device they run on.

It used to be hard enough to write portable applications and services for a relatively small number of device types (computers, mainly). It was even harder to port existing applications from one device to another, from one OS to another. Now imagine this task growing exponentially, together with the explosion of device types being invented these days (devices such as cell-phones, PDAs etc). By benefiting from a common framework, maintenance and development that is independent from hardware will be a much more gain than it would have been in the past.

Historically, developers built applications by integrating local system services. This model gave developers access to a rich set of development resources and precise control over how the application would behave. However, developers have already largely moved beyond this model. Today, developers are building complex n-tier systems that integrate entire applications together from all over their networks and then add unique value on top. This enables developers to focus on the functionality of the application rather than on building infrastructure. The result is a shorter development time, higher developer productivity, higher integration and, ultimately, higher-quality software.

However, there is no unifying technology that brings these development resources together. The closest thing to a unified infrastructure is the Internet; but this doesn't imply the existence of a unified development resource. .NET is important to developers because it will both change the way the development process takes place. Code developed with .NET technology will benefit fully from the other services and applications available on the Internet in their applications. One could even say that this would be the birth of a first Internet-scale operating system.

In order to accommodate as many developer classes as possible, Microsoft is planning on offering support for most programming languages available out there. This means that the transition to the .NET development will be seamless for most programmers.

What's the different from what we have today?

This discussion is best started by the words of Charles Fitzgerald, Director of .NET Platform Strategy:

"It's kind of interesting to go back and look at how we build software over probably the last ten or twenty years. And we really have kind of harnessed ourselves to this economic engine of growth known as Moore's Law, where the code we wrote this year, next year is going to run faster, is going to run more cheaply. And that's been a great phenomenon for really Microsoft and other companies to hitch their business to. And sort of the moral equivalent of having the wind at your back in terms of the set of investments you make. And we now live in this world where there actually are now two very powerful economic engines. Moore's Law still has at least five to ten years left on it, but there's another very powerful economic force, which is the plummeting cost of communications, and obviously we've seen a phenomenal amount of investment going to building up the Internet over the last five or six years. And we really are in a position where we have global connectivity. It's very, very low cost relative to everything that came before us. And the question we really started to ask was, how do you start to build really interesting applications in an environment where you have both abundant processing and also abundant communications capabilities. And at the very highest level, a lot of what we're trying to do with .NET, is figure out how to build a platform that harnesses both of these economic forces. We want to have the equivalent of the wind at our back, and also be running downhill. That's sort of where we start as we think about how to package software in a .NET environment."

.NET is about services. This new idea of software as a service and how do people build, deploy, operate, integrate, aggregate, consume this new set of software services that are delivered over the Internet.

Current technologies like IAS and ASP enabled the server-side functionality; however, they are different from .NET in the sense that they are particular and singular technologies that are not integrated.

When looking at what's happened over the last five or six years, there's really a revolution in the way users interact with applications, in the sense that users can now sit down, type in a URL, click on a link, go visit tens of millions of different sites. Going one step further, given this global connectivity, .NET will revolutionize the way applications talk to applications, or the way software talks to other software.

This idea is based on the fact that Internet itself is extremely distributed. This is the inherent nature of Internet; historically, it was built to preserve connectivity when one site is bombed during the war, the rest of the sites can still communicate with each other. Therefore, the Internet is a very distributed architecture.

Many applications these days do not take advantage of a distributed architecture, in the sense that we do all of our processing on the server, and we pretty much use all the power at the edge of the network as a terminal. And that terminal model, where you do all the processing on the server and periodically you ship out a picture of what happened on the server, when the user wants to do something, you send everything back to the server, that model really made sense in a world where processing power was scarce.

However, there's abundant processing power around the edge of the net. In 1999, Intel shipped on the order of two trillion MIPS. That's a phenomenal amount of processing power, most of which is sitting out on the edge of the network.

The question now, is, given the fact that communications fabric of the Internet is distributive, how to start to build applications in a distributed and very Internet native fashion? The end point is the idea of .NET, starting to build things as Web services, where users can still have very rich software content, but also take advantage of that near ubiquitous communications capability that currently exists in the world.

Moreover, .NET is more than just taking a frame based site, and one of the frames is another website that you fetch information from their website and make use of it. One of the challenges with HTML is it doesn't really give you information. It gives you a picture of the information. .NET wants to go beyond content aggregation, beyond screen scraping, and allow people to start programmatically build applications that stitch these different things together. Today, developers can build things on top of screen scraping, but it turns out to be very brittle. Once the site changes, the information that used to be meaningful may not be meaningful anymore.

For example, your bank has a Web site with some of your information. I can build an application today that goes out and screen scrapes those different sites and pulls the information together, and allows you to look at your complete financial situation. The problem with that solution is it's very brittle. The reason is the following. It might be the case that one day one might grab their account balance out of their bank's Web site; the next day after the bank gets bought or it redesigns its Web page, one may find oneself actually grabbing a different account number. So briefly, I may think I'm doing really well financially, but it makes for a very brittle mechanism on top of which to build these interactions between different systems. And .NET wants to have something that gives developers a real solid programmatic infrastructure to build on top of.

How does .NET accomplish these goals ?

At the core of the new development paradigm is the concept of a **Web service**. Web services are discoverable services in the sense that one can query what kinds of services are provided out there. Once these services are discovered, one can query their functionality. All this is achieved over the Internet through the Simple Object Access Protocol (SOAP), which is based on Extensible Markup Language (XML).

Conceptually, developers integrate Web services into their applications by calling Web application programming interfaces (APIs) just as they call local services (application or system calls). This is the reason for which the web will be viewed as an Internet-scale operating system. Instead of making the service calls to the local machine, programs will do it across the Internet to a service residing on a remote system. For example, a service such as Microsoft Passport could enable a developer to provide authentication for an application. By programming for the Passport service, the developer can take advantage of Passport's infrastructure and rely on Passport to maintain the database of users, make sure that it is up and running, backed up properly, and so on.

.NET is founded on this principle of Web services, and Microsoft is providing the infrastructure to enable this evolution to Web services through each of the pieces of the .NET platform. The next generation of development tools and infrastructure, including Visual Studio.NET, the .NET Framework, Windows.NET, and the .NET Enterprise Servers, have been designed for developing applications on the Web services model. The .NET Building Block Services, the new

.NET device support, and the forthcoming .NET user experience will provide the remaining pieces of the puzzle to enable the development of applications that take full advantage of the Web services model.

.NET Applications

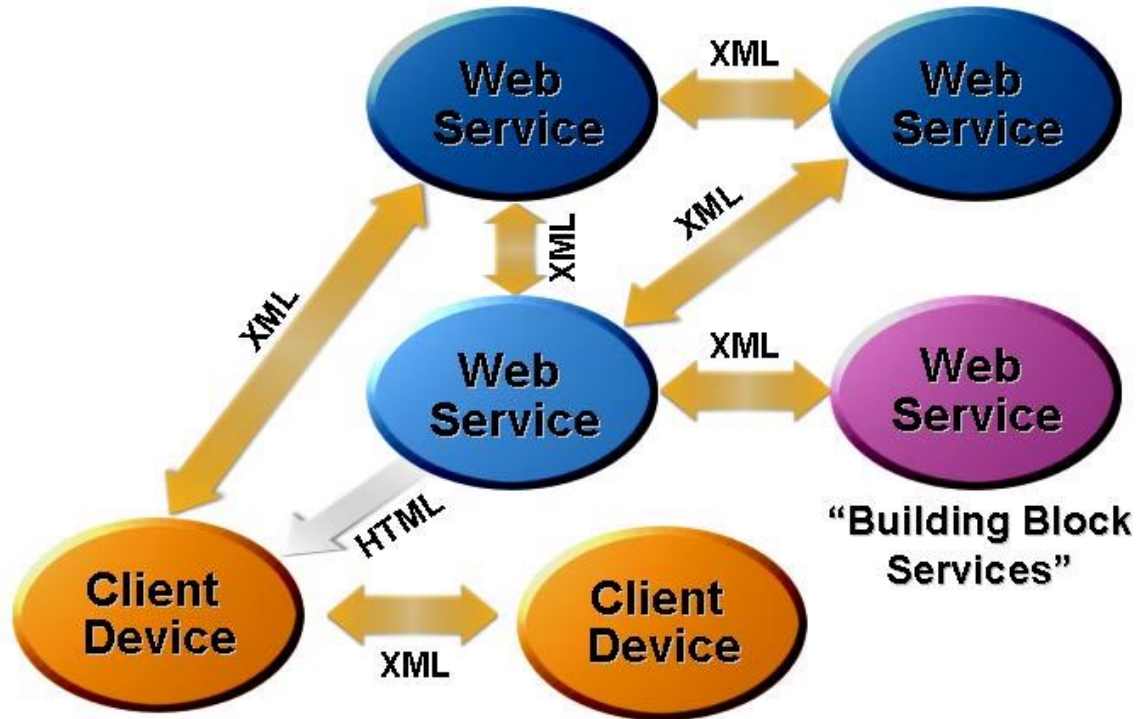


Figure 1. .NET Applications

By making use of the services provided on the Internet, developers will have the opportunity to share their work through components. Following the concepts used with products such as Delphi and Cbuilder components, Java classes as well as ActiveX, the idea of web components is taking shape. Developers will be able to make use of these components in a unified and OS-integrated way.

.NET Device Vision

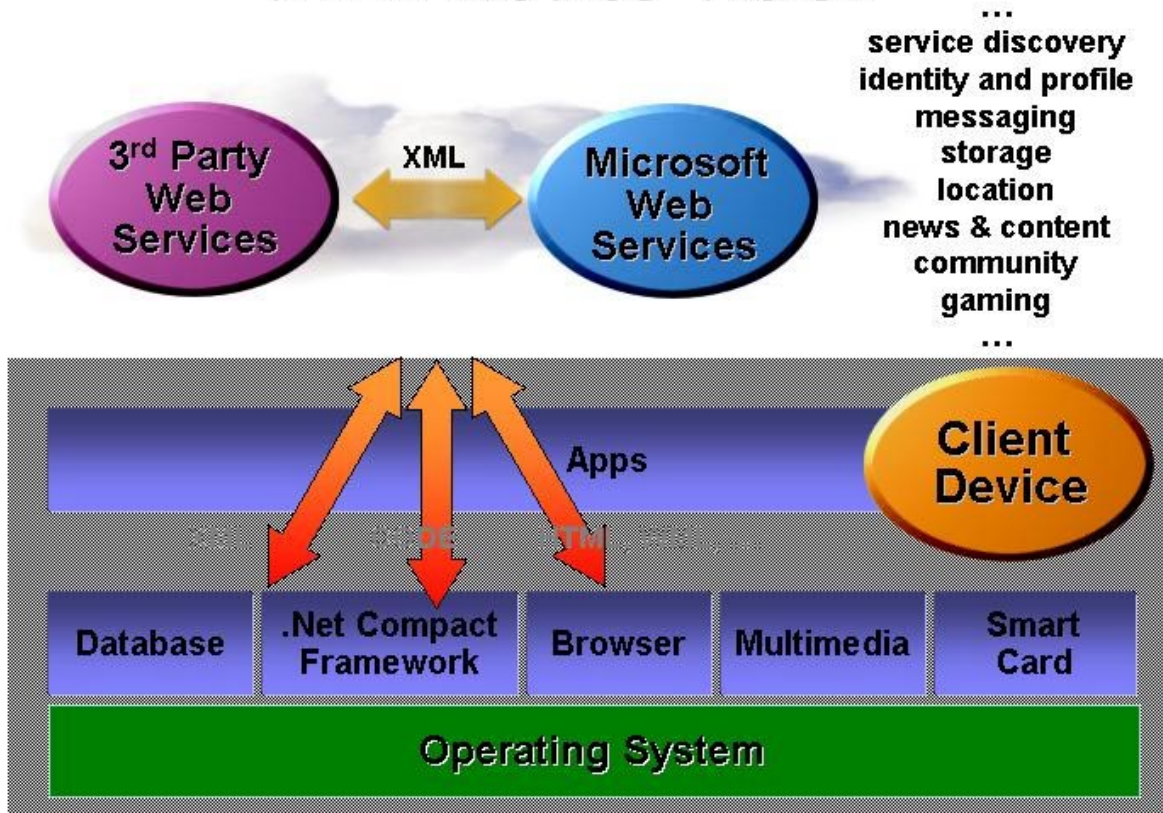


Figure 2. .NET Device Vision

In order for these goals to be achieved, Microsoft proposes an architecture of this development framework. This environment contains a set of building block services for the Internet operating system. For example Passport.NET will be used for user authentication. Other services will include anything that an OS normally offers: file storage, user preference management, calendar management, and many others.

For the developers, infrastructure and tools will be provided. Among the most important tools are Visual Studio.NET (a .NET extension of the already-popular Visual Studio suite, already in use by an estimated 6 Million developers), the .NET Enterprise Servers, the .NET Framework, and Windows.NET

Another .NET internal is CLR (Common Language Runtime). Basically, CLR is a runtime component responsible for code execution, security, component plumbing and dependencies. CLR will support most programming languages that exist out there (the list contains about 20 languages that are aimed at porting, including the most popular ones such as Perl, Jscript, C++, Java, SmallTalk). This will make sure that developers will find it easy to switch to this framework.

Inside the .NET Framework

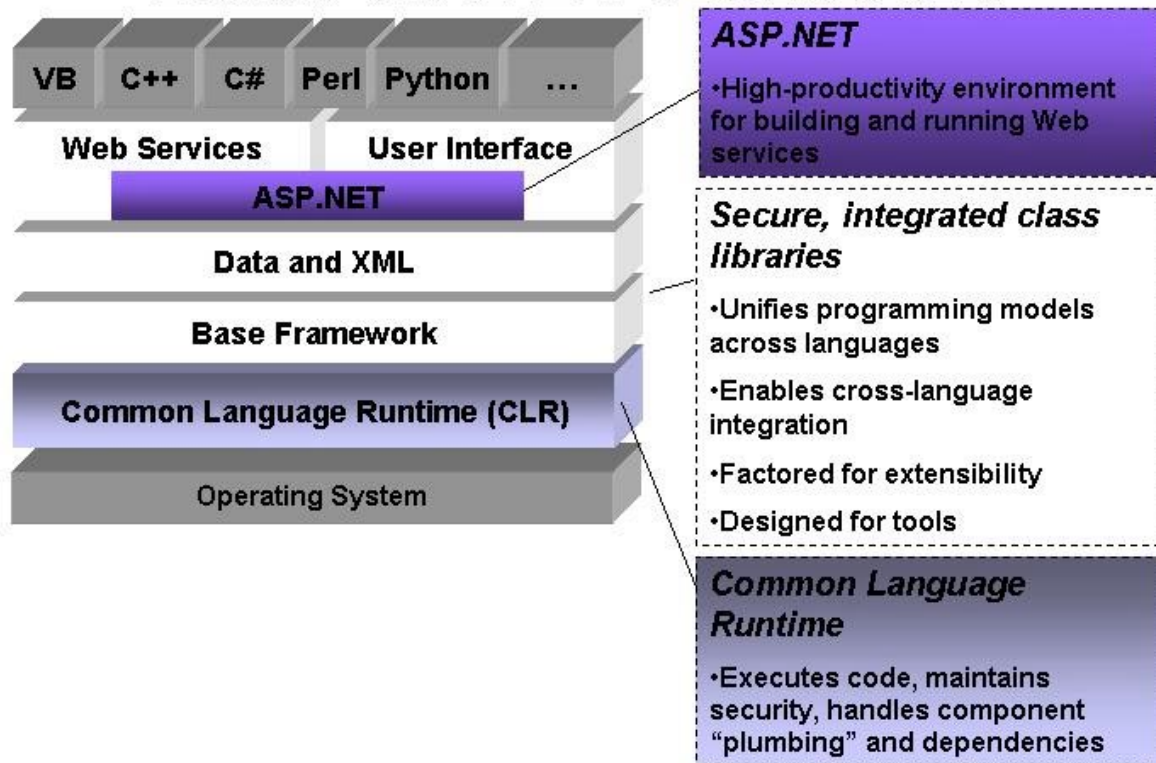


Figure 3. Inside the .NET Framework

A special framework will be provided for small devices developers. This framework is called .NET Compact. The architecture is lightweight while compatible with its desktop counterparts. The same development tools can be used for development. Another goal is to make it work across multiple OS's (not only Windows).

What do we already have?

As a reminder, there are three conditions that must be met in order to make the next generation of distributed computing a reality:

1. Everything needs to be a Web service, so that it can participate in the connected network. This applies to both applications and to resources like storage.
2. Web services must be easy to aggregate and integrate so that developers can quickly and efficiently create applications.
3. Users must have a simple and compelling experience working with Web services so that they will adopt the new applications and services.

In order to meet these conditions, Microsoft has developed some tools to enable developers start building the new software paradigm.

.NET Framework and Microsoft Visual Studio.NET developer tools.

These two tools are used for developing software and web services. The .NET Framework is a multi-language component development and execution environment.

Windows and the Microsoft .NET Enterprise Servers. Windows® and the .NET Enterprise Servers

These two tools are designed to aggregate and deliver Web services. Servers like Windows 2000 Server Family, Microsoft Exchange 2000 using XML as its core. The use of XML ensures that data will be unique among applications and vendors.

.NET Foundation Services.

One of the principal aims of .NET Foundation Services is to make .NET-enabled user experiences both simple and compelling. In order to enhance user experience, Microsoft has built services like identify, notification that enables users to switch from one service to another and decrease the user's overhead when switching (e.g. Re-login).

Device software.


As one of the philosophies is to access any device, Microsoft is building services for devices that people uses in their everyday lives. Microsoft is engaged in building software for everything from phones to PDAs, to whole other sets of devices like the tablet PC.

.NET-compatible applications.

Microsoft is not only investing in the infrastructure for .NET, but it is also developing applications built on that infrastructure. These applications integrate Web services with their own functionality to deliver a targeted user experience. For consumers, Microsoft is building the MSN network of Internet services.

Below is a big picture of the tools and services that are currently available or will be available by 2002.

.NET Platform



	2000	2001	2002+
Infrastructure and Tools	.NET Framework, Visual Studio.NET tech previews	.NET Framework, .NET Compact Framework, Visual Studio.NET	v2
Building Block Services	Passport	Additional key services	Corporate federation
Enterprise Servers	.NET Enterprise Servers in beta	.NET Enterprise Servers	v2
Operating Systems	Windows 2000	Windows XP	Blackcomb

Figure 4. The present and future of .NET

.NET concerns and issues.

.NET sounds promising in the sense that it can provide a better computing experience to users. However, some concerns arise with the .NET vision.

First of all there is the privacy. As stated above, user data are all stored on the Internet. The data includes personal information, health information, bank statement, VISA number, and other confidential information. It is then a big issue of the privacy of the information. Which company can store the information? Microsoft or any other company that the user can choose? Can government access this information? Can a user erase all these information permanently? What kind of mechanism ensures that no hackers can steal the information? Is web security all that secure? A user using a centralized data system would have these concerns and all .NET services must address these problems.

The ability to communicate between services is one of the main points of .NET. This point suggests that each service has some form of openings for other services to consume. Security mechanism must then be used in order to ensure correct use of services. Issues like a service trying to hack into other service must be addressed.

A third issue is about market acceptance. .NET will revolutionize the way people use and develop software. The revolution is so great that it would take a lot of time for the market to swallow and for developers to accommodate this new developing strategy. As an illustration, a bank may be using a certain system for more than 20 years. The bank has spent millions of dollars to develop that

system 20 years ago. Would the bank dump its old system and start to develop a system using .NET, or should it rewrite all its code ? Would the cost of transitioning to .NET outweigh the benefit? If the benefit were not greater than the cost, most business would not bother switching to .NET.

A fourth issue is about networking. Currently people access the Internet using 56K or broadband. In either case, there is still not enough bandwidth to transfer large amounts of data. One philosophy of .NET is that it is an "Internet wide Operating System." Thus, it is not hard to imagine that the size of the services that would be transmitted over the network would be large. Additionally, if everyone is using .NET services, the network could be overloaded. Possible solutions to this would be either an advance in networking technology or distribution of server software in many places.

The last issue is dealing with hardware failure handling. What would happen if a network were down? Since data and services are all on the network, users cannot access their data in this situation. Similarly, not even the Operating System will work. Caching data and services locally could solve this.

Conclusion

.NET represents a significant evolution from current approaches to computing. While in a design phase at this point, it will most certainly represent a big step ahead in the way software is developed and used. Since it targets the new generation of mobile gadgets, it will probably shape the future of its software. The design of the .NET framework will seriously ease the use and development of distributed applications. We expect that the .NET technology will become one of the most popular development environments and probably the most used technology for consumer software consumers.

References:

- Microsoft .NET website, <http://www.microsoft.net>
- Jamie de Guerre, Microsoft Research. "The .NET Framework". Microsoft Presentation at University of Alberta, 2001